链滴

Dragonite -- 一个基于 UDP 的传输层（Java）

作者：linker

原文链接：https://ld246.com/article/1536392924641

来源网站：链滴

与KCP相比，Dragonite更快，也更适合Java生态。

```
public final class EchoMain {

    public static void main(final String[] args) throws SocketException, InterruptedException {
        val parameters = new DragoniteSocketParameters();
        parameters.enableWebPanel = true;
        parameters.webPanelBindAddress = new InetSocketAddress( port: 8001);
        val dragoniteServer = new DragoniteServer( port: 9225,  defaultSendSpeed: 102400, parameters);
        DragoniteSocket tmpDragoniteSocket;
        while ((tmpDragoniteSocket = dragoniteServer.accept()) != null) {
            val dragoniteSocket = tmpDragoniteSocket;
            print("New connection from ${dragoniteSocket.remoteSocketAddress.toString()}");
            new Thread(() -> {
                try {
                    while (dragoniteSocket.alive) {
                        val bytes = dragoniteSocket.read();
                        print(new String(bytes));
                        dragoniteSocket.send(bytes);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    try {
                        dragoniteSocket.closeGracefully();
                    } catch InterruptedException | IOException | SenderClosedException ignored {
                    }
                    print("${dragoniteSocket.getRemoteSocketAddress().toString()} connection closed");
                }
            }).start();
        }
    }

    private static void print(final String msg) {
        System.out.println(msg);
    }

}
```

上面的例子，比较简单，形式上看Dragonite和传统的TCP API是类似。

每个UDP虚拟连接都开了一个新的线程来处理，在UDP虚拟连接数低于20个时候是合适的。

如果需要支持1000+以上的UDP虚拟连接，最好是用一个线程池来处理，配合定时器。

Dragonite为了支持很多链接的情况，提供了multiplexer，也就是多路复用器。

这样就可以用回掉的方法来处理从不同连接来的UDP Frame而不需要开太多的线程。

multiplexerAcceptThread.start();

这句是启动了 多路器的核心线程，想理解多路器原理的读者可以看下。

```
val multiplexerAcceptThread = new Thread(() -> {
    try {
        muxHandler.run();
    } catch InterruptedException | MultiplexerClosedException e {
        if dragoniteSocket.alive {
            Logger.error(e,  message: "Cannot accept multiplexed connection");
        }
    }
},  name: "FS-MuxAcceptor");
multiplexerAcceptThread.start();

byte[] buf;
try {
    while (buf = dragoniteSocket.read()) != null {
        multiplexer.onReceiveBytes(buf);
    }
} catch InterruptedException | ConnectionNotAliveException ignored {
} finally {
    try {
        dragoniteSocket.closeGracefully();
    } catch final Exception ignored {
    }
    multiplexer.close();
    Logger.info("Client "{}" ({}) disconnected",
            infoHeader.name, dragoniteSocket.remoteSocketAddress.toString());
}
```

原文链接：Dragonite -- 一个基于 UDP 的传输层（Java）